ServeSense Coding Standards (v1)

Scope: This guide codifies how we write, organize, document, and test code in the ServeSense Django project. It mirrors the structure of the "Coding Standards by Saba" document and adapts it to our current codebase.

---

1. Consistency

- Follow the same patterns across apps (`menu`, `reservations`, `staff`, `tables`).
- Prefer one idiom per task (e.g., function-based views for CRUD in these apps; class-based views only when they add value).
- Keep naming, imports, and formatting uniform across the repo.

  Tools: Use **Black** (line length 88), **isort**, and **flake8**. Run them before committing.

---

2. Naming Conventions

2.1 Packages & Modules

- **Packages (directories)**: all lowercase, short names.
- **Modules (files)**: all lowercase with optional underscores (e.g., `forms.py`, `views.py`, `urls.py`).

2.2 Django Apps

- App names are lowercase, singular where possible: `menu`, `reservations`, `staff`, `tables`.

2.3 Models

- **Classes**: `CamelCase` (e.g., `MenuItem`, `Reservation`, `Staff`, `Table`).
- **Fields**: `snake_case` (e.g., `best_seller`, `reservation_time`).
- Include `verbose_name` / `verbose_name_plural` and `__str__` where appropriate.

```
class MenuItem(models.Model):
    name = models.CharField(max_length=100)
    price = models.DecimalField(max_digits=6, decimal_places=2)
    available = models.BooleanField(default=True)
    best_seller = models.BooleanField(default=False)

    def __str__(self) -> str:
        return self.name
```

2.4 Views

- **Function-based views**: `snake_case` (e.g., `menu_list`, `menu_edit`, `menu_delete`).
- **Class-based views** (when used): `CamelCase` + `View` suffix (e.g., `ReservationListView`).
- Use action-oriented names that describe behavior.

2.5 Forms

- **Classes**: `CamelCase` ending with `Form` (e.g., `MenuItemForm`).
- **Fields**: `snake_case`.

2.6 URLs & URL Names

- URL paths: lowercase words separated by hyphens when meaningful (`"tables/"`, `"edit/<int:pk>/"`).
- URL **names** (for `{% url %}` reversing): `snake_case` verbs+nouns (e.g., `menu_list`, `table_edit`).

```
urlpatterns = [
    path("", views.menu_list, name="menu_list"),
    path("edit/<int:pk>/", views.menu_edit, name="menu_edit"),
    path("delete/<int:pk>/", views.menu_delete, name="menu_delete"),
]
```

2.7 Templates & Static Assets

- Templates live under each app: `<app>/templates/`.
- Template filenames: lowercase with underscores (e.g., `menu_list.html`, `edit_reservation.html`).
- Static assets: group by feature; use lowercase-hyphen filenames (e.g., `reservation-list.css`).

---

3. Comments & Documentation

- Use **Google-style docstrings** for modules, classes, and functions.
- Keep docstrings focused on **what** and **why** (the code itself shows the **how**).
- Add inline comments sparingly for non-obvious logic or business rules.

```
def menu_list(request):
    """Display all menu items and handle creation.

    Args:
        request (HttpRequest): Incoming request.

    Returns:
        HttpResponse: Rendered list or redirect after successful form
```

```
submission.
    """
    ...
```

4. Formatting & Indentation

- Indentation: **4 spaces**.
- Max line length: **88** (Black default). Break long expressions over multiple lines.
- Use trailing commas in multi-line literals and calls to improve diffs.
- One statement per line; no extraneous whitespace.

5. Error Handling

- Fail fast for missing records: use `get_object_or_404(Model, pk=pk)`.
- Validate input using Django Forms or ModelForms; check `form.is_valid()` before saving.
- Catch **specific** exceptions; avoid bare `except:`.
- For user-facing flow, prefer redirects with the Django messages framework over printing/logging to the console.

```
obj = get_object_or_404(Table, pk=pk)
try:
    obj.delete()
except ProtectedError as exc:
    messages.error(request, "Cannot delete table in use.")
    return redirect("table_list")
```

6. Imports

- Group imports: 1) stdlib, 2) third-party (Django), 3) local.
- One import per line; avoid wildcard imports.
- Use absolute imports for intra-project references.

```
# stdlib
from decimal import Decimal

# third-party
from django.shortcuts import render, redirect, get_object_or_404

# local
```

```
from .forms import MenuItemForm
from .models import MenuItem
```

7. URL Formatting (Project-Level)

- In `ServeSense/urls.py`, keep app includes grouped and commented.
- Root path maps to a clear home view (currently `reservations.views.home`).
- No trailing slashes in URL names; ensure namespace clarity if apps add duplicates.

8. Template Style

- Use a shared `base.html` with `{% block %}` sections.
- Always include `{% csrf_token %}` in modifying forms.
- Prefer semantic HTML; keep inline styles to a minimum.
- Escape variables by default; only use `|safe` for trusted content.

```
<form method="post">
  {% csrf_token %}
  {{ form.as_p }}
  <button type="submit">Save</button>
</form>
```

9. Code Readability

- Keep functions small and cohesive; extract helpers for repeated logic.
- Use meaningful names; avoid cryptic abbreviations.
- Prefer early returns to reduce nesting.

10. Code Reusability

- Share common form patterns via base forms/mixins.
- Reuse templates with `{% include %}` and template inheritance.
- Consolidate duplicated view logic into utilities or class-based views where it simplifies code.

11. Testing & QA

- Use `pytest` + `pytest-django` or Django `TestCase`.

- Test layers: models (constraints, `__str__`), forms (validation), views (status codes, redirects), URLs (reverse resolution), templates (context keys).
- Maintain **≥80%** coverage; block merges below threshold.

```python
from django.test import TestCase
from django.urls import reverse

class MenuViewsTests(TestCase):
    def test_menu_list_renders(self):
        url = reverse("menu_list")
        resp = self.client.get(url)
        self.assertEqual(resp.status_code, 200)
        self.assertContains(resp, "Menu Management")
```

---

12. Security

- **CSRF**: Always include tokens for POST forms.
- **XSS/Injection**: Use Django ORM and template auto-escaping.
- **Authentication/Authorization**: Protect staff/admin endpoints; use decorators like `@login_required` or permission checks.
- **Secrets**: Never commit `SECRET_KEY`, DB creds, or API keys. Load from environment.
- **DEBUG**: `DEBUG=False` in production; configure `ALLOWED_HOSTS`.

```python
# settings.py (excerpt)
SECRET_KEY = os.getenv("DJANGO_SECRET_KEY")
DEBUG = os.getenv("DJANGO_DEBUG", "0") == "1"
ALLOWED_HOSTS = os.getenv("DJANGO_ALLOWED_HOSTS", "").split(",")
```

---

13. Logging & Settings

- Configure `LOGGING` in `settings.py` with at least console handler and INFO level in prod.
- Separate settings by environment (e.g., `settings_dev.py` vs `settings_prod.py`) or via env flags.
- Use Django `settings` only inside functions to ease testing/mocking.

---

14. Database & Migrations

- One migration per feature/PR; keep them small and descriptive.
- Run `makemigrations` and `migrate` locally before pushing.
- Avoid destructive migrations without data backups or staged rollouts.

15. Frontend (Static) Conventions

- Put static files under each app's `static/<app>/` directory.
- Name CSS/JS files by feature (e.g., `reservations-list.css`).
- Minimize inline scripts; attach JS at the end of the body or via bundler.

16. Documentation

- Maintain Sphinx docs under `/docs/`; keep them updated with public APIs and app-level guides.
- Cross-link modules using autodoc where feasible.

17. Git & Commit Hygiene

- Branch naming: `feature/<slug>`, `fix/<slug>`, `docs/<slug>`.
- Conventional commits are encouraged: `feat:`, `fix:`, `docs:`, `refactor:`, `test:`, `chore:`.
- Open small, focused PRs with clear descriptions and screenshots (for UI changes).

18. Pre-commit Hooks (Recommended)

Add a `.pre-commit-config.yaml` with:

- `black`
- `isort`
- `flake8`
- `djhtml` (optional, for templates)

19. Example Lint/Format Config

`` (excerpt):

```
[tool.black]
line-length = 88
target-version = ["py311"]

[tool.isort]
profile = "black"
known_first_party = ["ServeSense", "menu", "reservations", "staff", "tables"]
line_length = 88
```

`` (excerpt):

```
[flake8]
max-line-length = 88
extend-ignore = E203, W503
per-file-ignores =
    */migrations/*: E501
```

---

20. Checklist (Before Commit)

-

Version: 1.0\ **Applies to**: ServeSense Django project and all first-party apps.\ **Owners**: Team ServeSense (update alongside architectural changes).